Beowulf - a DBMS alternative for GLAST

With thanks to Richard Fink,

Code 631

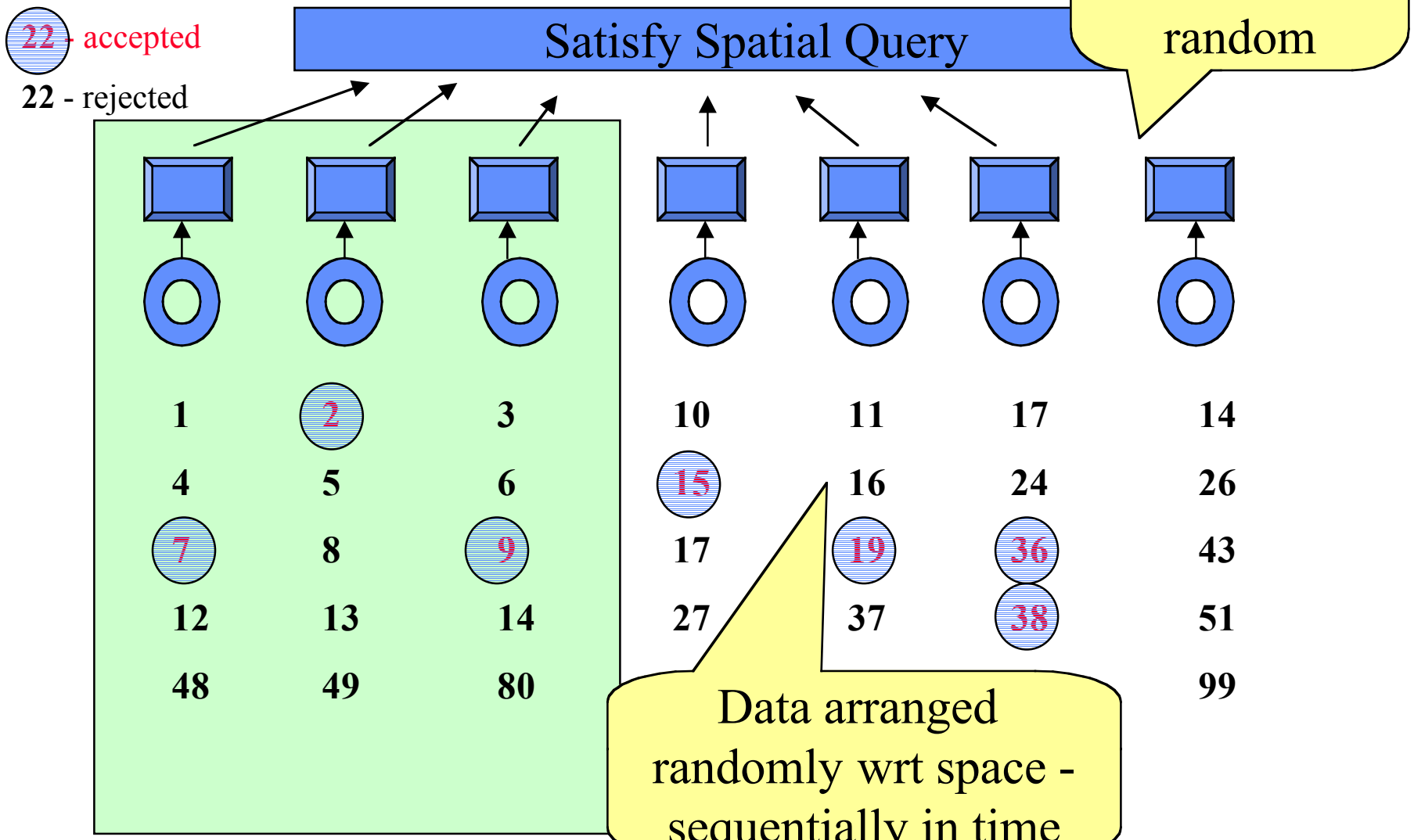
Spatial Organization

- The GLAST study report recommended narrow-field FITS files each of which represents a mosaic tile on the sky.
- Reprocessing would mean reassignment of photons to other FITS tile files and their deletion from their current location.
- Queries involving indices that were NOT spatial could be very slow, e.g. find all photons > 20GEV
- BUT spatial queries are fast because they are sequential.
- Update in place of a particular photon would be hard.
- Commercial DBMS systems that manage photons in the FITS files would be expensive - and the update and time-order retrieval problem still exists.

Beowulf

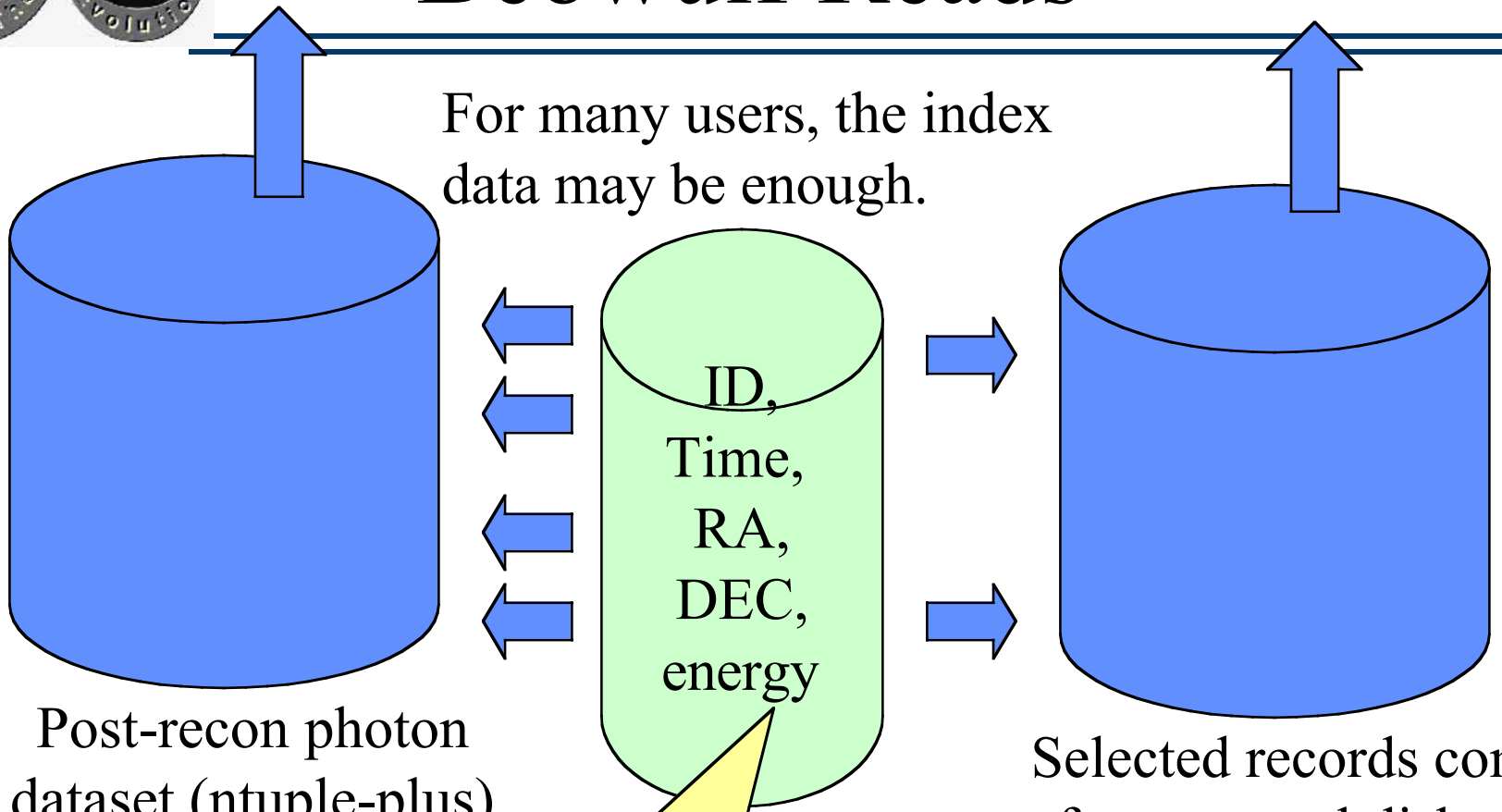
- Beowulf creates supercomputers out of commodity PCs for low price/high performance.
- With disks on each PC, the Beowulf has a large number of independent and parallel I/O channels.
- Setting each PC to read the photons on its disk and accept/reject them maximizes throughput.
- Storing data randomly in space and sequentially in time is an efficient I/O strategy for GLAST
- Avoiding high cost DBMS licensing is a plus

Beowulf


22 - accepted
22 - rejected
Satisfy Spatial Query
Thruput is max when required elements are random
1
2
3
10
11
17
14
4
5
6
15
16
24
26
7
8
9
17
19
36
43
12
13
14
27
37
38
51
48
49
80
99
Data arranged randomly wrt space - sequentially in time

Beowulf Reads

For many users, the index data may be enough.



Post-recon photon dataset (ntuple-plus) is read randomly based on index

Index file of minimal records is read sequentially.

Selected records come from several disks - maximizing thruput

The Index Table

Random reads are costly in I/O. High-yield queries will be faster read sequentially. All queries will have an initial index-read overhead. The index file for 1 year is ~2GB - feasible for keeping in memory.

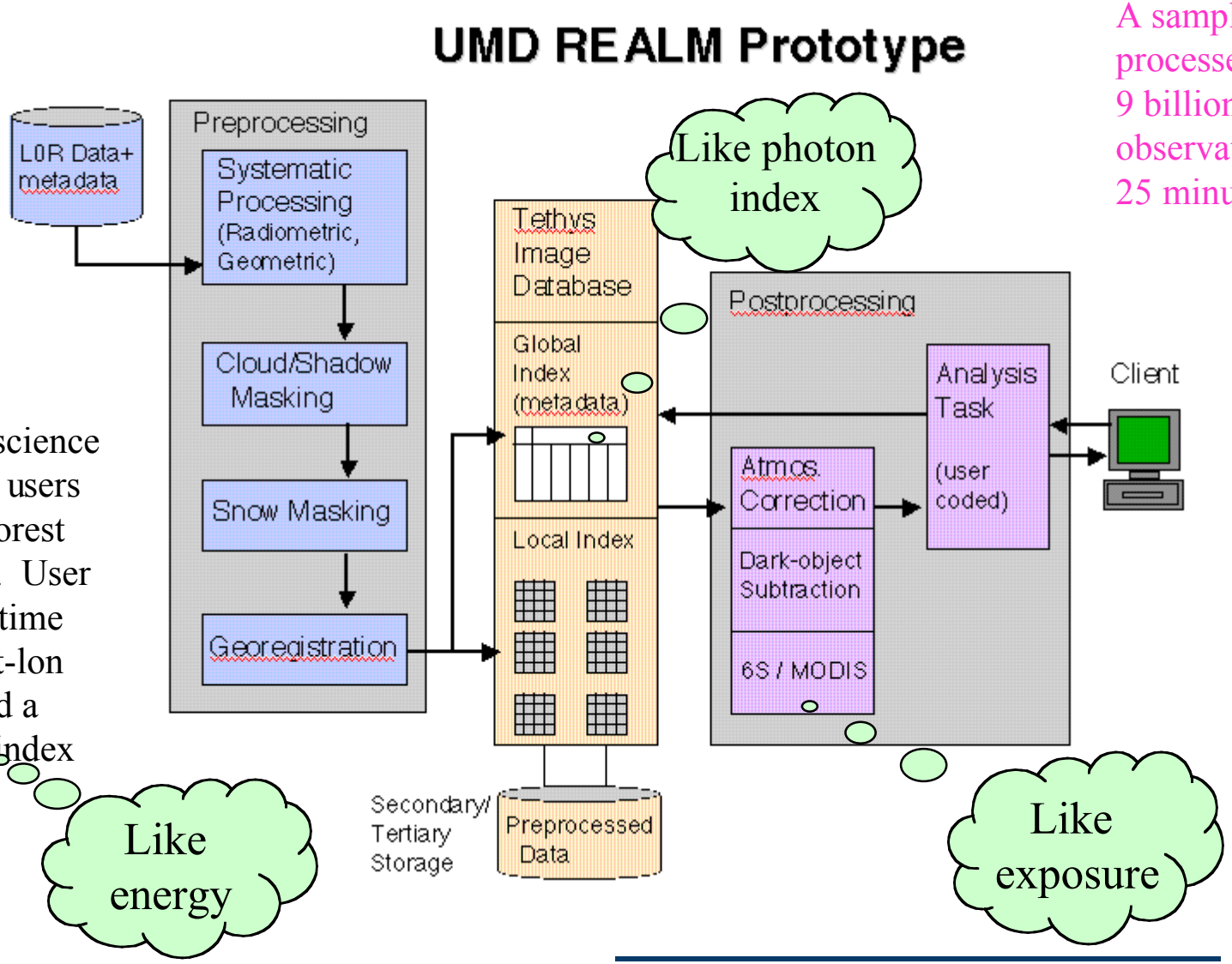
Time in secs.

Events per Access	*Index Read Time*	*Event Read Time*	*Ethernet Transfer Time*	*Total time*	*Archive Service Rate (per day)*
1,000	150	9	0.02	159	543
10,000	150	84	0.2	234	369
100,000	150	833	2	985	87
1,000,000	150	1680	20	1550	55
10,000,000	150	1680	200	1730	50

The entire index must be read, no matter how few photons are actually retrieved

Once the index is read and the yield is high, it is faster to read the data file sequentially than randomly

An example



A sample query processed nearly 9 billion separate observations in 25 minutes

In this earth science application, users request a forest cover image. User requests a time range, a lat-lon range and a vegetation index range.

Upgrade Path

- Add entire computers and disks over time without changing the architecture.
- Replace components if they break
- Not have to rewrite any software
- Requires the choice of a platform and operating system that has a lot of inertia, i.e. likely to persist
- Requires an architecture that is extensible.
- Hardware replacement can be done with a distributed DBMS or Beowulf - but license cost may be a big issue with a commercial DBMS